AFRL-RI-RS-TR-2014-170

A TOOL FOR COMPLIANCE AND DEPTH OF DEFENSE METRICS

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

JUNE 2014

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2014-170 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

// SIGNED //
JEFFREY DEMATTEIS
Work Unit Manager

// SIGNED //
WARREN H. DEBANY JR
Technical Advisor, Information Exploitation & Operations Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JUNE 2014	FINAL TECHNICAL REPORT	AUG 2012 – DEC 2013

4. TITLE AND SUBTITLE

A TOOL FOR COMPLIANCE AND DEPTH OF DEFENSE METRICS

5a. CONTRACT NUMBER
FA8750-12-2-0233

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

David M. Nicol

5d. PROJECT NUMBER
DH52

5e. TASK NUMBER
UI

5f. WORK UNIT NUMBER
LL

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Illinois at Urbana-Champaign
Coordinated Science Laboratory
1308 West Main Street
Urbana, IL 61801-2307

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RIGA
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2014-170

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This project developed a software tool NP-View that analyzes the configuration files of firewalls and routers to determine the connectivity that is permitted, and identify connections that violate global policy. It uses the discovered connectivity as a basis for computing metrics that address the degree to which the network is resilient to stepping-stone attacks. Features of the tool were strongly influenced by the results of pilot studies we performed with potential users.

15. SUBJECT TERMS
Networking, security, firewall, compliance

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**JEFFREY DEMATTEIS**
U	U	U	SAR	22	19b. TELEPHONE NUMBER (*Include area code*) **N/A**

TABLE OF CONTENTS

Section **Page**

LIST OF FIGURES

Figure **Page**

1 SUMMARY

"A Tool for Compliance and Depth of Defense Metrics" sought to transform a research project (NetAPT) into a commercially viable software tool (NP-View) with metrics oriented output. Both tools analyze configuration files from actual network devices, discover the topology of the system so described, perform a connectivity analysis, and determine whether the flows which are permitted are compliant with some machine checkable global policy description. The project began with a code base that grew over several years as a graduate student's research project and was limited to one old model of one firewall. We proposed to transform NetAPT into a commercial (and maintainable) tool, by refactoring the code and algorithms, and by working closely with potential customers to identify and features needed for its anticipated use in federal compliance evaluation of electric utilities. The project achieved all of its technical aims, and the new tool NP-View is now licensed by a startup company that grew out of this effort, Network Perception.

2 INTRODUCTION

Electric utilities that generate and transmit bulk electric power are subject to Federal regulations developed and enforced by the North-American Electric Reliability Corporation (NERC) for Critical Infrastructure Protection (CIP), known more simply as the NERC CIP requirements. These regulations are aimed at ensuring that utilities operate in ways that enhance the reliability of electric power. Some of the standards address the cyber infrastructure. These are built around the notion of "critical cyber assets", which are cyber devices whose failure or compromise impacts the overall system reliability. The so-called "Electronic Security Perimeter" (ESP) can be thought of as a ring of protection devices (like firewalls) that surround all critical cyber assets, and adjudicate connections between critical assets and devices on the other side of the ESP. The NERC CIP requirements require that all passage through the ESP be documented, and that it be demonstrable that all possible connections to a critical asset be known a priori and that all ports and services available on a critical asset likewise be documented.

The state of the practice in NERC CIP audits is for auditors to be presented with manually generated network diagrams, with the ESP denoted. Auditors ask to see the documentation for the protection of a critical asset, and then review the actual configuration files of the firewalls that limit access to that device, looking for the rules that guard it. This approach is labor-intensive, selective in scope, and error-prone. Much of what's involved here can be automated. In particular

- Topology discovery and visualization of the network.
- Computation of the network flows that are permitted by the system's access control rules.
- Identification of unexpectedly permitted connections to critical assets.

In addition, the concern for cyber-security makes attractive analysis features that aren't necessary for NERC CIP compliance demonstration, but follow naturally from the NERC CIP focus on connectivity.

The end product of this project is a commercially available tool "NP-View", marketed by a start-up company Network Perception. NP-View is highly focused on support for NERC CIP audits. NP-View is useful both to prepare utilities for audits and to sustain detailed understanding of their network's connectivity and security posture, and to aid auditors in their assessments. The contract supported transition of a demonstrated technology in the form of a tool NetAPT into one that was prepared for commercialization. To make this transition we had a number of technical challenges and objectives, which are outlined in the subsections to follow.

2.1 Topology Inference and Rule Recognition

Topology discovery in NetAPT leveraged an open source tool, Antfarm, developed by Sandia National Laboratories. Antfarm synthesizes various pieces of connectivity information into a more comprehensive whole. Experience with Antfarm on large topologies highlighted the need to investigate alternative approaches. Experience showed that the time spent discovering topology grew to be a factor of ten or more larger than the time needed to perform a connectivity analysis. Performance instrumentation showed us that Antfarm's reliance on the MySQLite database software was the root cause of the problem----most of the topology discovery time was spent in system I/O, manipulating the one large file used during that computation. For us to meet our goal of scaling to large networks, we had to find another way to discover the topology.

2.2 Refactored Analysis Algorithms

The core analysis algorithm in NetAPT looped over every known host in the model. For the selected host it performed a depth-first traversal of the entire network with the host as root, looking for the hosts that are reachable. As part of this traversal, firewall rules are applied as the firewalls are encountered. We saw the need to refactor this algorithm, in two ways. First, NERC CIP requirements are interested in accessibility of **all** hosts on the other side of the ESP, not just hosts that are discovered from the topology. We also saw an opportunity for performance optimization, as sometimes the interest is in whether it is at all possible for one network to connect to another *at all*, not to find all connections that are permitted. We couldn't get there from the NetAPT algorithm.

2.3 Multiple Vendors Supported

NetAPT, using Antfarm, was able to parse and interpret the formatting of Cisco firewall rules (of a particular, now rather old model) as well as those of iptables. To have impact in the electric utility sector we had to be able to parse configurations from a larger set of firewall vendors, and have the analysis engine support the various differences that exist in firewall manufacturer

world-views and implementation of filtering policies. Fairly serious effort was going to be needed to expand the applicability of our approach.

2.4 Incorporate Routing

NetAPT made no assumptions about how flows are actually routed. Pushing a potential flow into a firewall, it assumed that whatever got through the ingress routes might leave through every other port in the firewall. Of course, real networking doesn't work that way, and the result of this assumption was that NetAPT would find and report many many more paths than the actual system would permit, because the actual system would not push a flow through anything but the interface described in the forwarding table. The new tool would have to integrate routing into its analysis fabric.

2.5 Connectivity Metrics

While metrics don't play a role in NERC CIP evaluation, the larger use of the tool to help maintain knowledge of how well a network is protected (and the fact that the DHS program category is interested in metrics) led us to propose development and implementation of connectivity metrics. Our initial thought was to measure the degree of compliance of network flows with global policy, but later discussions with potential customers suggested another approach to us. In any case, NetAPT had no substantive support for metrics, and these were needed.

3 METHODS, ASSUMPTIONS, AND PROCEDURES

We saw that success was going to depend on (i) our being able to rewrite much of the code NetAPT used, (ii) making sure our resulting software focused on the needs of potential customers, and (iii) develop documentation and training materials for the new tool.

Our approach for software development was to lay out a development road map and track progress on that using project planning software that manages goals and milestones. We always had a good idea of where we were in the development process. The algorithm redesign was discussed among the developers prior to implementation, but individual developers took the responsibility for design and implementation changes, in different parts of the system.

As proposed, we worked closely with several pilot industrial partners in the course of development, and staged several releases of the tool with those partners. This was invaluable in providing industrial configurations to stress test the tool's parsing and analysis, and to identify features of the tool that should be on our development roadmap. We have now over 30 beta testing users of the prototypes developed under this contract.

For documentation and training we utilized a professional technical writer employed by the University of Illinois to develop these materials. We also developed materials for a training short course, and held that course in conjunction with an electric industry meeting focused on coming changes in the NERC CIP requirements.

4 RESULTS AND DISCUSSIONS

We'll address the results following the same subsection structure of the Introduction.

4.1 Topology Inference and Rule Recognition

The automated interpretation of configuration files requires one to formally *parse* those files. This means that for each type of file to be analyzed a description of what one expects to see (also known as a *grammar*) must be developed. As pieces of a configuration are recognized by the program that compares the input against the supposed grammar, one can interpret elements of the configuration. For example, different firewall vendors use different formats to describe a firewall rule, yet rules contain pretty much the same information---what are the source IP addresses, what are the destination IP addresses, what are the source and port ranges involved, which protocols are permitted? For a given type of configuration, its parser can, upon recognizing (for example) a source IP address range, the parser code can record that source IP address in a device independent way. In like fashion, different vendors will express information about their devices' network interfaces in different ways, but when recognized by a parser, the information that is essentially common to all interfaces can be extracted and recorded. Figure 1 illustrates the logic of the inference engine, accepting the configurations of different types and models of devices, storing in canonical form the configurations discovered, and then producing for analysis two files. One file describes the firewalls and the rules they contain, the other file describes all elements of the network.

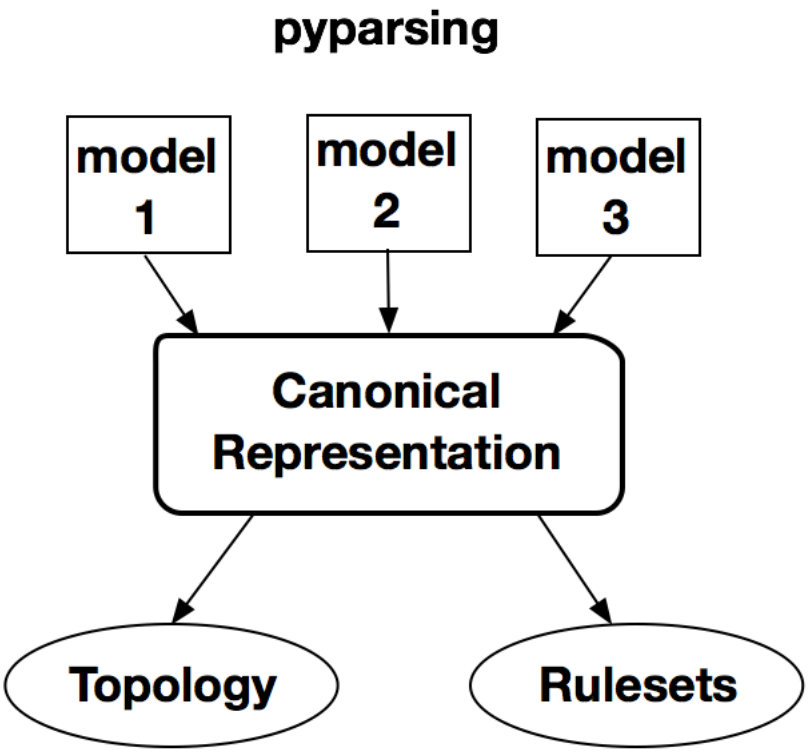

Figure 1. NP-View Configuration Processing

Our project developed a new topology discovery algorithm. It is built around a simple set of rules.

1. Identify all networks faced by known device interfaces (so called *internal* networks). Topologically, two devices that face a public network with exactly the same description are able to communicate directly through that network.
2. Identify all address blocks referenced by rules, group references, and routing statements within the configuration files. Address blocks that are hosts within or sub-ranges of some internal network are attached to that network.
3. Process routing statements, identifying for each the address block being routed, the network faced by the interface through which the route is directed, and the gateway IP address serving as the nominal destination for the routed traffic. If the gateway IP is not an interface for some existing device, then create a gateway node attached to the egress network, and associate with that node the address block being routed.
4. Any address blocks not found to be internal or attached to any gateway are considered to be external.

Classification of networks is an important element of the selective analysis NP-View performs, to be described later in this report.

After parsing configurations and classifying networks, NP-View creates a visual display of the network, such as that shown in Figure 2. The NP-View user can interact with this display, moving elements around, customizing colors (to denote various categorization), expanding and contracting the hosts attached to a network, and pulling up via dialogs other information about selected elements of the network.

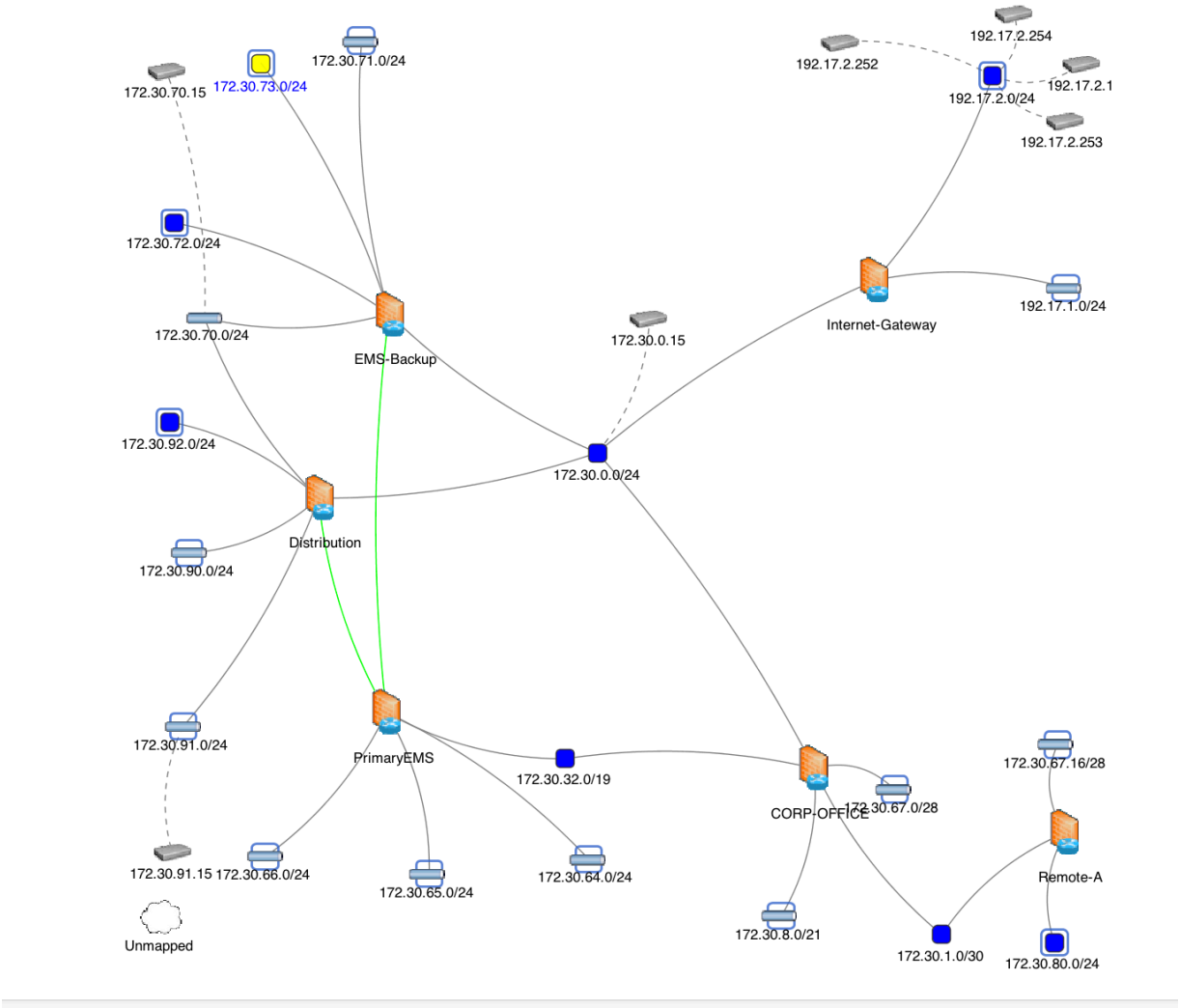

Figure 2. NP-View Topology Display

4.2 Refactored Analysis Algorithms

The core of the analysis algorithm is to work through desired combinations of source and destination address blocks. By looking at address blocks rather than individual hosts, we may be able to identify collections of flows (known as "flow-sets" that are permitted between two networks with one analysis, rather than working through all combinations of hosts, reducing the complexity of the computation.

In the discussion to follow we refer to a range of contiguous IP addresses as an "address block".

Given a source address block S and a destination address block D, we define the notion of a "flow-set" to be the collection of all connections permitted between some host whose IP address is in S with some host whose IP address is in D. The algorithm follows the topology and routing rules to find the path between S and D that the network uses, and then computes the related flow-set. Starting from S, the analysis visits every firewall (or router) with an interface in the broadcast domain that contains S. Knowing the interface through which the flow would enter and the destination address block, processing first consults routing information to determine which exgress interface(s) are involved. It then identifies the rules that govern passage between the ingress and exgress interfaces such that there is overlap between the rules' source IP address and the S and overlap between the rules' destination IP address and D. This is a super-set of rules that might permit passage of some flow from S to D, and if that superset is empty there is no further exploration through that device. A non-empty super-set is saved for later processing.

When path exploration reaches D we have identified a sequence of firewalls between S and D, and for each a set of rules that might allow passage. We then work through combinations of the rules in these supersets, in an order equivalent to what would be applied in the actual system. For a given selection of rules the operations are of intersection in a four dimensional space, as follows. Each rule describes a number of objects, each with a range of source IP address and range of source port address, a range of destination IP address and destination port addresses. For a given sequence of rules we compute the intersection of objects represented by the rules, as any flow from S to D is contained within the intersection. We have to be aware though that some of the members of this intersection may represent flows that would be admitted by sequences of rules earlier in the order, meaning that in the real system an affected flow would be admitted by a different sequence of rules than the one being analyzed. This is important, because NP-View needs to provide the correct set of rules that admit a flow, not simply that a flow is admitted. The set of objects that correspond to flows admitted by the rules sequence is obtained as the set difference of the intersection of rule objects with the union of objects represented by rules that appear earlier in the processing order. With a small bit of additional filtering focused on making sure that the right protocols are included, each of these objects is a flow that the configuration permits.

Different analyses can be initiated from the graphical user interface, using this baseline approach of finding flows between a given source S and destination D. These include

- Selecting a source (alt., destination) network through the user interface and ask for all flows that are permitted with that network as source (alt., destination).
- Select a rule, or access-control list, or device from the user interface and ask for all flows that are admitted by that rule, by some rule in the access-control list, or by any rule in any access-control list in the device.
- Select a virtual private network tunnel from the display, and ask to see all flows that cross that tunnel.
- Define groups of networks, and initiate analyses to identify all flows between networks in the source group to networks in the destination group.

For each of these cases, NP-View first determines the set of address blocks that need to be considered as sources and as destinations to identify the required flows, and performs the computation between all pairs of selected source and destination address blocks. The results of an analysis are provided graphically, and in tabular form. Figure 3 illustrates the path of one flow that is permitted (here from 172.30.8.20 to 172.30.64.10), Figure 4 illustrates one tabular view of that flow, which includes identification of the firewall rules involved. Figure 5 illustrates a spreadsheet form which can be exported, annotated in a notes column and re-imported into the project.

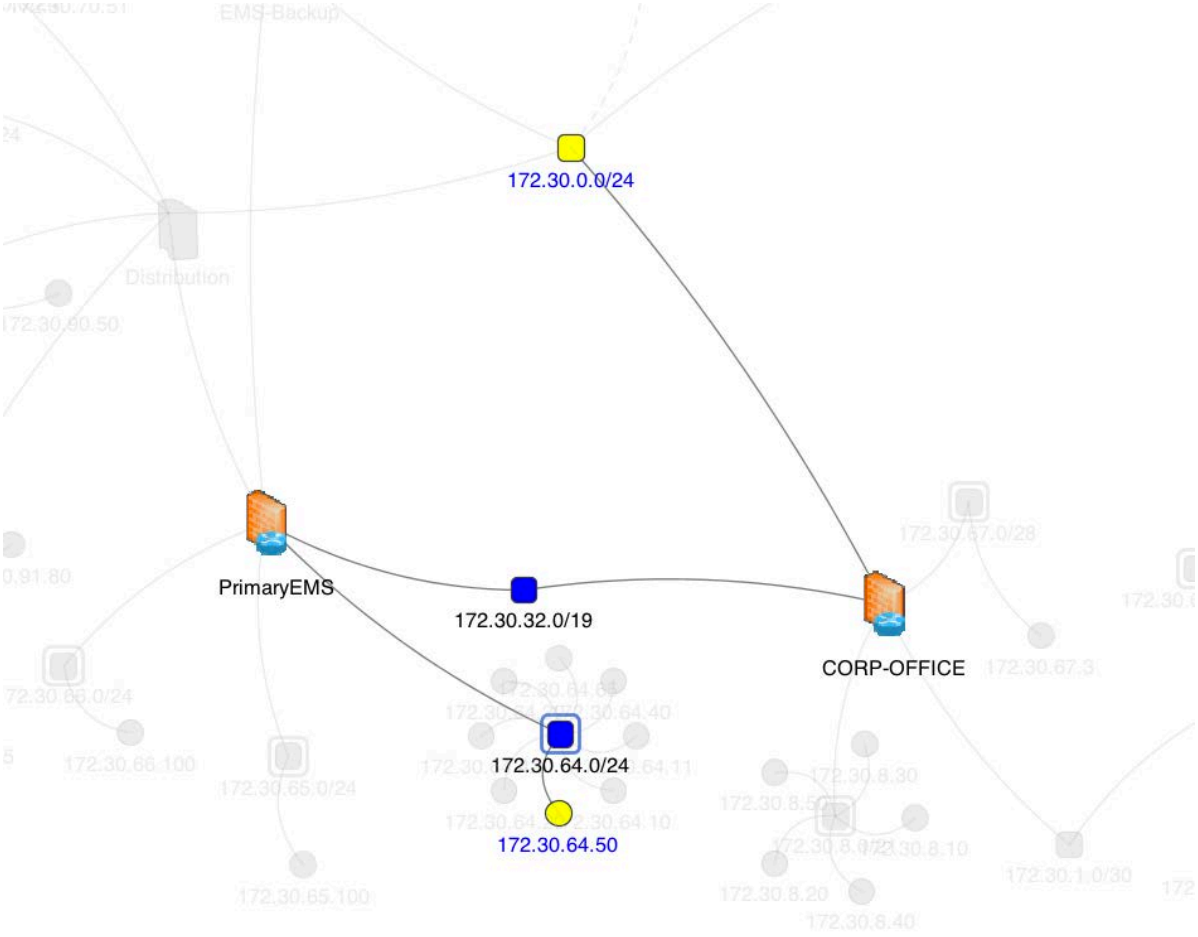

Figure 3. NP-View Illustration of Identified Flow

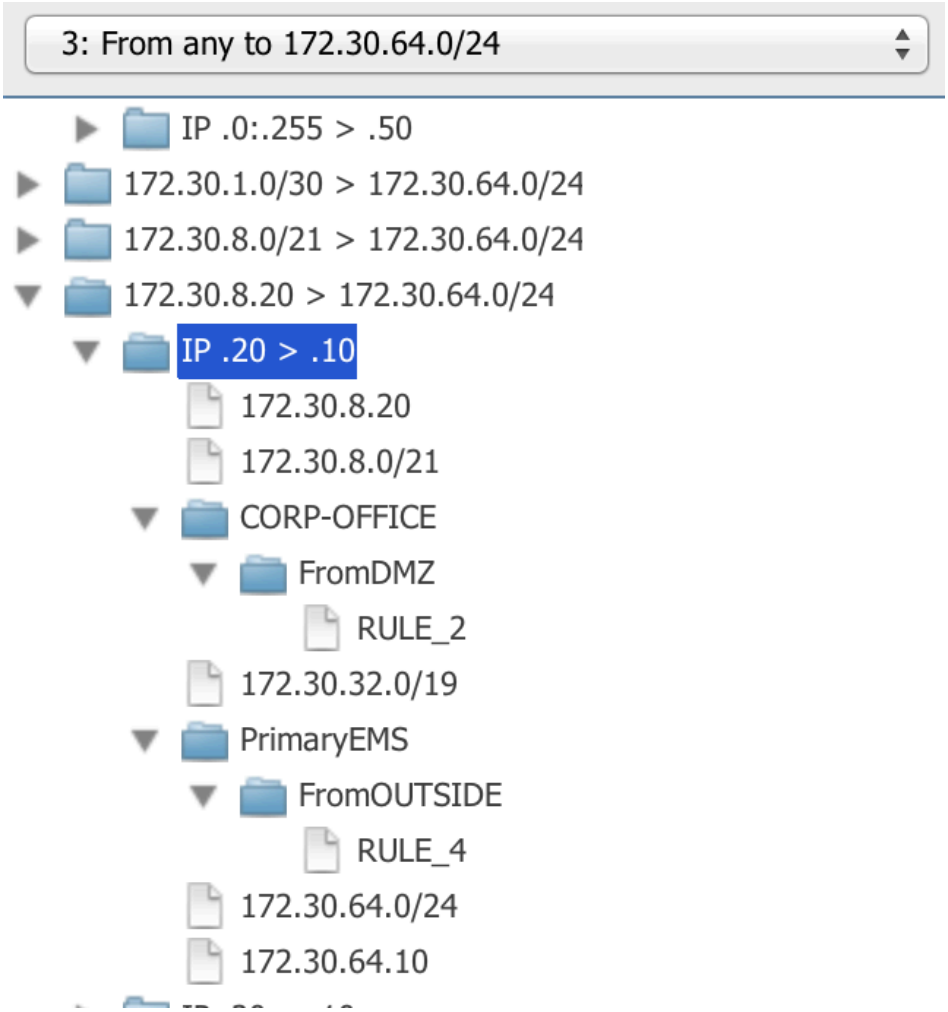

Figure 4. NP-View Tabular Display of Path

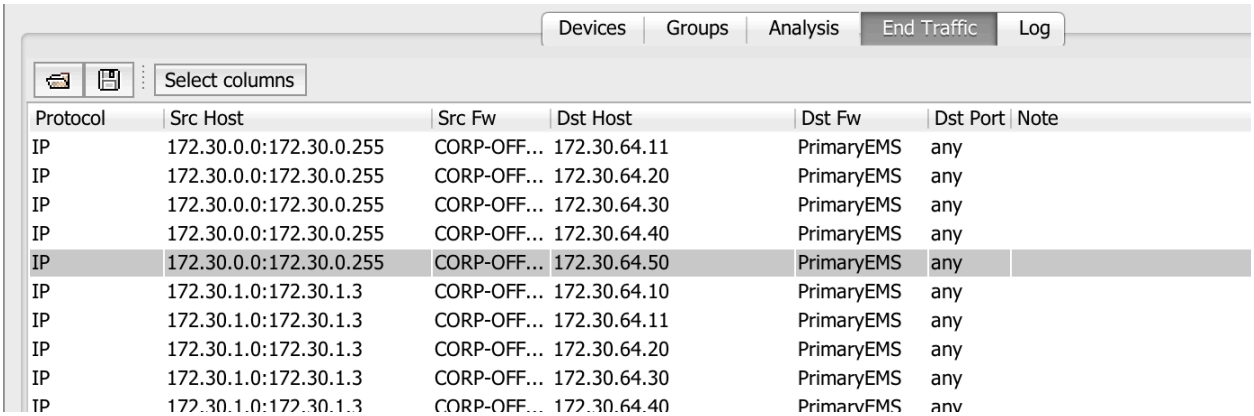

Figure 5. NP-View Spreadsheet Display of Paths

Another performance optimization that proved to be critical is so-called "Fast Path" processing. Under a FastPath analysis the exploration for flow-sets between a selected source address block S and destination address block D halts as soon as *some* flow between them is discovered. FastPath processing cuts out a considerable amount of computation, and yields higher level information about connectivity between networks. We have found from industrial partners that of greatest interest is when FastPath shows that there is *no* connection between a pair of networks, which, when expected, shows that the configuration correctly separates them.

4.3 Multiple Vendors Supported

Firewalls made by different vendors do pretty much the same thing, however there are differences that matter and must be represented within NP-View. For example, Cisco 8.4 and later admits traffic to virtual private networks differently than other vendors, and different vendors apply network address translation in different ways.

We developed the approach of representing at a block diagram level the sequence of processing of a flow set through a device. The analysis engine knows the vendor and model for the device through which the flow set is passing and uses for a given step a block of code that is correct for that device at that step. A significant number of these steps use the same code block for all supported devices, which reduces the management overhead over developing and maintaining completely different code sets for every vendor and model.

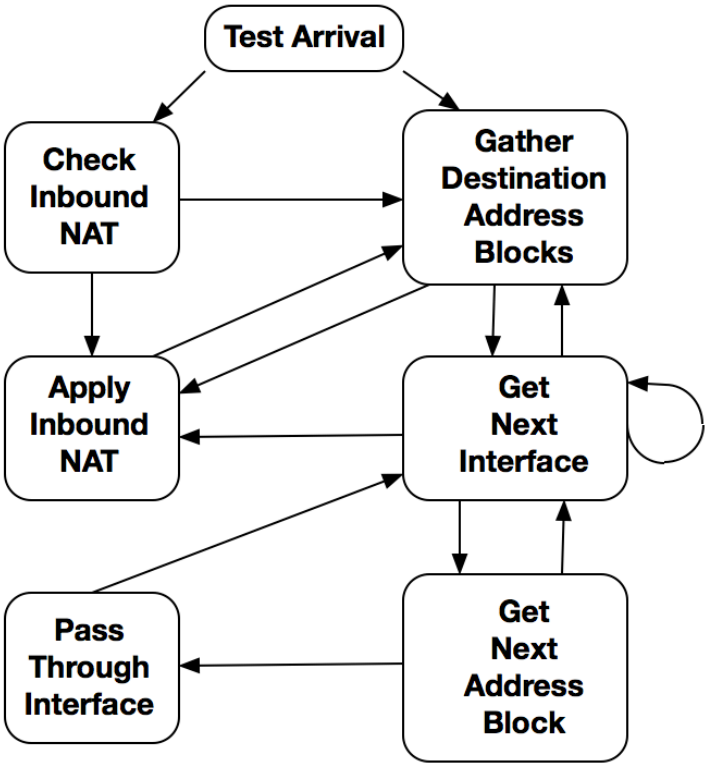

Figure 6. Processing of Flow-set Through Device

Figure 6 illustrates the seven basic steps of the analysis. At a high level (and with details specific to the devices supported) these blocks perform the following functions.

1. *Test Arrival*. Build a cache of rules at this device which overlap in the source addresses with the source network S. See whether the ingress interface might require network address translation (NAT), and branch accordingly.
2. *Check Inbound NAT*. A comparison of the incoming destination address space with NAT rules is performed, and if there is a match, a list of the destination address blocks that result from the translation is produced and remembered. If such a transition occurred processing is directed to the block '*ApplyInboundNAT*', otherwise it is directed to *Gather Destination Address Blocks*.
3. *ApplyInboundNAT*. The 'next' destination address block from the transformed list is obtained, if any. If every destination block in the list has already been processed, then processing of the flow set through this device is completed.
4. *Gather Destination Address Blocks*. Use the destination network D to look up from the forwarding table the interfaces to which flows from S might be routed. Routing may induce fragmentation, in which case D may fragment into sub-addressblocks as a function of the way D is routed. In the absence of the source being NAT translated, this

block is "home" for the rest of the processing, in that it will be responsible for knowing when all of the processing associated with pushing S to D flows through the device has finished. It does this by maintain a list of the address blocks that were reported as a result of consulting the forwarding table, pushing one source/destination pair out at a time, with control ultimately being returned back once that pair has been completely processed. If the call to this code block finds the list of source/destination pairs exhausted, control is returned to *ApplyInboundNAT* if the original inbound address block was NAT transformed, and otherwise processing at this device, for the original source/destination pair is completed.

5. *Get Next Interface.* This block identifies the 'next' device interface through which the flow might traverse, using information obtained from the forwarding table. If the list of interfaces left to process is empty, control is routed back to *Gather Destination Address Blocks* for (potential) further processing. Alternatively, should another egress interface be found, the known ingress and egress interfaces it identifies the rules that govern this transition (if any) and whether this transition actually requires rules (they are not when all that is required is routing). If rules are required but no rules that overlap the source and destination address blocks under processing are found, control loops back to this same block to pick up the next interface to process. Otherwise, control branches to *Get Next Address Block*.
6. *Get Next Address Block*. The next pair of source/destination address blocks to process is examined, and if the source address block matches any NAT rules for the selected egress interface, the transformation is applied (potentially creating more source/destination address block pairs to process). If there is another pair, control passes to *Pass Through Interface*, otherwise control returns to *Get Next Interface.*
7. *Pass Through Interface*. The selected (and potentially fragmented and transformed) source and destination address blocks are examined against firewall rules that might admit them. If no such rules exist, processing returnes to Get Next Interface. Otherwise, the pair are passed either over a virtual private network (VPN) tunnel or into the logic that handles flow entries in a network.

4.4 Incorporate Routing

Routing is dynamic. Sometimes one finds some so-called static routes declared within a firewall configuration, but most routes are established by the execution of some routing algorithm. We incorporate routing information into our models in two ways. First, when static routes are declared within a firewall configuration file, we parse that and include the route in the data structures for the firewall. Second, for devices whose routes are established largely by dynamically executed algorithms, we parse the output of commands one applies to the firewall or router to report which routes are present, and include these into the model. This particular feature was driven by our interaction with an industrial partner whose network contains many dynamically configured routers.

4.5 Connectivity Metrics

An NP-View analysis can identifies all the connections permitted by a system's configurations. For each pair (hs,hd) of hosts where hs is the source and hd is the destination (and they reside in different broadcast domains) NP-View identifies the port ranges and the protocols involved in possible connections. This provides the basic connectivity information. The stepping-stone metrics we develop consider the impact of so-called stepping-stone attacks, where hosts are compromised and used essentially as switches. A sequence of these gives an attacker access to hosts he cannot otherwise directly access. The metrics of interest relate to the sensitivity of a configuration to stepping stone attacks, in terms of numbers of stepping stones required, and numbers of stepping stone attacks that are possible.

Given source host hs and destination host hd, we can compute the minimum number of stepping-host hosts that must be compromised in order for an attacker on hs to reach hd. This computation is easily performed using breath-first-search. A second metric measures, for each number of hops h, the number of unique stepping stone sequences of length h that give an attacker on hs access to hd. This is a measure of *width*, with a large width indicating that many different attack vectors exist from hs to hd.

The dual of vulnerability analysis is 'criticality' analysis. Rather than select a host or network and ask about stepping stone attacks *towards* the selection, we can ask about the number of stepping stone attacks that can launched *from* the selection. This gives a measure of how critical a host or network is from the point of view of defense. For, if many short stepping stone attacks can be launched from the selected host or network, it becomes an attractive asset for an attacker to acquire.

We have in NP-View integrated computation and visualization of these metrics. Figure 7 illustrates an example.

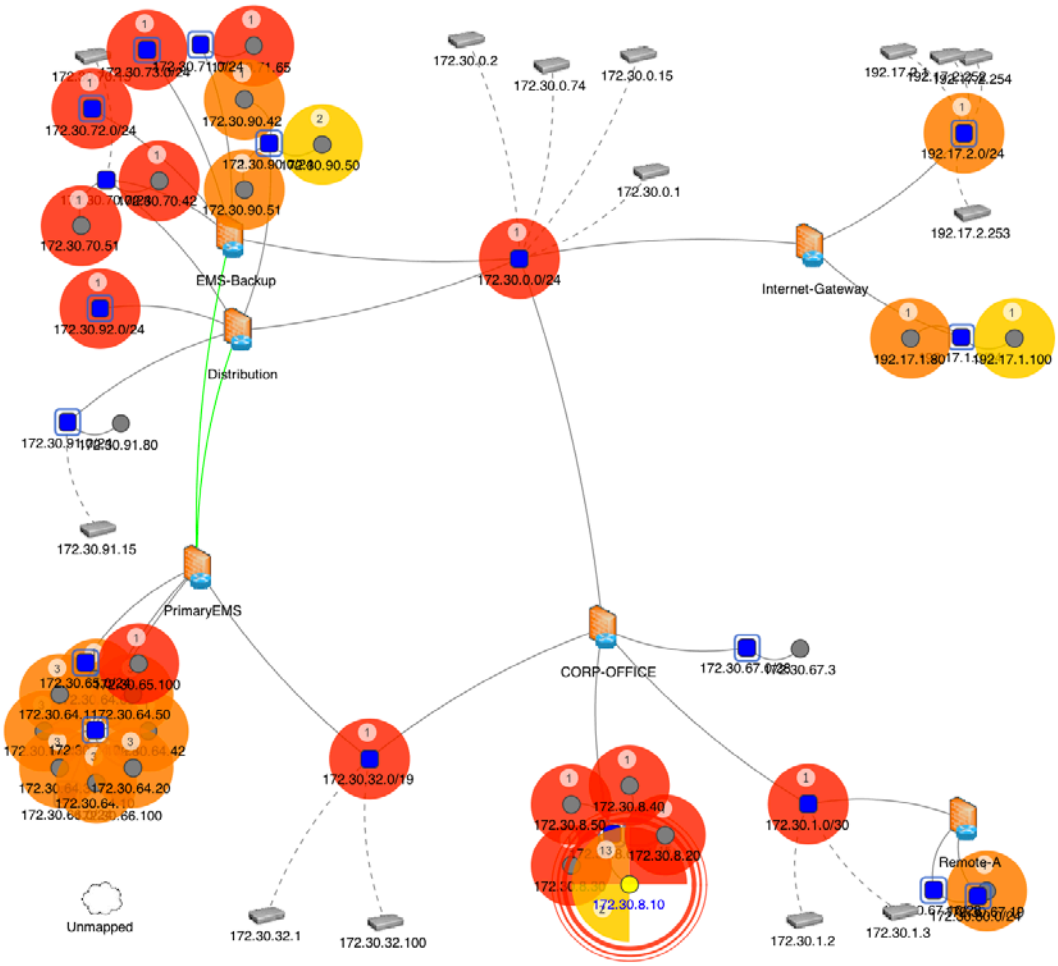

Figure 7. NP-View Visualization of Stepping Stone Attack Locations

Here the host 172.30.8.0 (at the bottom center) was chosen as the attack target. Hosts that can reach this target directly without any compromise are highlighted in red. Hosts that can reach the target with a minimum of 1 stepping stone are highlighted in orange, hosts that can reach the target with a minimum of 2 stepping stones are in yellow. Each colored host is annotated with the number of unique stepping stone attacks that can be launched from that host.

Since the conclusion of the contract we have extended this capability further. We've augmented NP-View to take the results of scans performed by the vulnerability assessment tool nmap, and augment the topology with hosts not discovered already by analysis of the configuration. We now can look up the known vulnerabilities of open services discovered by nmap, and annotate hosts with their vulnerability scores from the National Vulnerability Database (NVD). We then use this information to discover the stepping stone paths that are easiest to accomplish (as a function of the exploit difficulty scores that have bee read in). We've discovered that under

realistic assumptions about the costs of vulnerabilities in stepping-stone attacks, the computational complexity of finding the most easily accomplished attack is NP-Hard, and so developed Monte Carlo based ways of exploring potential stepping stone paths to find the least cost ones. A paper "Modeling and Analysis of Stepping Stone Attacks" describing this work will appear in the proceedings of the 2014 Winter Simulation Conference [Nicol, 2014].

5 CONCLUSIONS

In line with the overall DHS program objectives, this project aimed to transform mature and promising prototype technology into commercial form. We did exactly that. A start-up company, Network Perception (see www.network-perception.com) now licenses NP-View. Thanks to DHS support we were able to transform NetAPT into a form with better, scalable algorithms, with a code base rewritten to support maintenance and extension, and a feature set that was significantly influenced by interaction with selected partners. We have placed NP-View with over 30 industrial beta-users. We are very grateful to DHS and AFRL for their support of this project.

6 REFERENCES

[Nicol, 2014] David M. Nicol and Vikas Mallapura, "Modeling and Analysis of Stepping Stone Attacks", *Proceedings of the 2014 Winter Simulation Conference*, Washington D.C., December 2014, to appear.

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

DHS	Department of Homeland Security
ESP	Electronic Security Perimeter
NAT	Network Address Translation
NERC CIP	North American Electric Reliability Corporation, Critical Infrastructure Protection
NVD	National Vulnerability Database
VPN	Virtual Private Network